SERENITYVAULT PROTOCOL · SÉRIE 8

Un pipeline vérifiable : classification, décision, preuve, audit

Vue d'ensemble technique du protocole, de ses composants, de ses règles de gouvernance non négociables, et de l'état exact de son implémentation.

Livre blanc technique · Série 8 · Stade actuel : ALPHA

ALPHA Ce document décrit une implémentation logicielle de test (ALPHA). Aucune cryptographie de production, aucune clé réelle, aucune donnée client n'est en jeu. Le protocole n'a fait l'objet d'aucune validation matérielle, terrain, juridique, institutionnelle ou réglementaire, ni d'aucun audit externe indépendant.

Objectif du protocole

SerenityVault Protocol définit un pipeline en quatre étapes — classifier, décider, prouver, auditer — dans lequel chaque étape est confiée à un composant distinct, et où aucun composant ne peut contourner celui qui le précède ou le suit.

L'objectif n'est pas de rendre une décision "infaillible", mais de garantir que toute décision sensible produit, au moment où elle est prise, une trace vérifiable indépendamment de l'implémentation qui l'a générée. C'est ce dernier point — la vérifiabilité indépendante — qui distingue une architecture de gouvernance d'un simple système de journalisation.

Les couches du protocole

TCAC — classification scellée et immuable

Chaque action entrant dans le système est d'abord classifiée selon un référentiel fixe à cinq catégories (ST, SN, CR, CS, INT). Le référentiel est versionné : toute modification produit une nouvelle version et une nouvelle empreinte cryptographique, de sorte qu'aucune classification passée ne peut être réinterprétée silencieusement a posteriori.

ADÈLE — moteur de décision déterministe

ADÈLE rend, pour chaque action classifiée, une décision parmi trois issues testables : `ALLOW`, `DENY`, `ESCALATE`. Le moteur ne peut être contourné par aucune action critique. Il produit une justification structurée et intègre un scanner de contradictions permettant de détecter les incohérences entre décisions. En cas d'ambiguïté, le refus (`DENY`) est la valeur par défaut.

METATRON — format de preuve scellée

Chaque preuve associée à une décision est un objet scellé par une empreinte SHA3-512. Aucune preuve ne peut contourner METATRON : toute altération, même minime, rend le scellement invalide et détectable.

Journal chaîné et racine de Merkle

Les décisions et les preuves s'accumulent dans un journal en ajout seul (append-only), chaîné par hachage. Le protocole calcule, sur les instantanés du journal, une racine de Merkle SHA3-512 déterministe et à séparation de domaine. Toute altération d'une entrée antérieure est détectable par recalcul de la chaîne et de la racine.

SAK v2 — rapports multi-profils avec rédaction

Le rapport SAK peut être généré selon trois profils distincts — auditeur, tribunal, régulateur — chacun avec ses propres règles de rédaction (redaction) des informations sensibles, à partir d'un même journal source.

Vérificateur Rust indépendant

Une seconde implémentation, écrite en Rust et indépendante de l'implémentation Python de référence, vérifie les preuves et les journaux. Elle est testée sur un jeu de fixtures de parité afin de garantir une concordance exacte avec l'implémentation de référence — un tiers peut ainsi vérifier une preuve sans faire confiance à une implémentation unique. Un exécutable Windows 64 bits est fourni pour la vérification hors ligne, sans installation de chaîne de compilation.

API SVP-X — Evidence / Audit

Une API sandboxée, décrite par un contrat OpenAPI 3.1, expose la vérification des preuves et des audits. Elle sépare les permissions vérificateur et auditeur, limite la taille des requêtes, et n'expose aucune route ni opération de Tier 1.

Némésis — compilation judiciaire supervisée

Némésis opère exclusivement en Tier 2 et sous supervision. Sa fonction est de compiler des faits déjà scellés, sous la spécification `ACT-AUD-02` ; il ne dispose d'aucun accès direct au Tier 1 et ne rend aucun jugement sur l'innocence, la culpabilité ou la vérité. Son implémentation ne peut aller au-delà d'un stub tant que la couche de preuve lisible par machine (A3) et SAK v2 (A4) ne sont pas pleinement fonctionnelles — une dépendance délibérée, pas un oubli.

Principes de gouvernance non négociables

01	Aucun secret réel — clés, adresses IP, identifiants, données client, contenu de coffre — n'entre jamais dans le système ; seules des valeurs de test sont utilisées.
02	Toute action critique doit produire un chemin de décision testable : `ALLOW`, `DENY`, ou `ESCALATE`.
03	Aucune action ne peut contourner le moteur de règles ADÈLE.
04	Aucune preuve ne peut contourner METATRON.
05	Aucun composant de Tier 2 n'accède directement au Tier 1.
06	Némésis est de Tier 2, strictement forensique ; il ne juge jamais.

07 Toute sortie du système distingue explicitement son stade de maturité : ALPHA, Beta, Release Candidate, ou Production-Proven.

08 Toute modification du référentiel TCAC crée une nouvelle version et une nouvelle empreinte.

Discipline de vérification

La feuille de route logicielle a progressé sous couverture de test croissante et sans régression signalée à aucune étape : 14 tests Python au premier assemblage fonctionnel, 56 tests Python à la livraison couvrant PR-001 à PR-012, complétés par une suite de tests Rust et sept jeux de preuves de parité entre les deux implémentations. La chaîne de compilation Rust n'a été introduite qu'après une autorisation explicite, conformément à une gouvernance de changement délibérée plutôt qu'à une évolution silencieuse. L'ensemble de la suite s'exécute hors ligne, sans clé, secret, ni accès réseau.

État de la feuille de route (PR-001 à PR-012)

ÉTAPE	CONTENU	ÉTAT
PR-001	Squelette du dépôt	Complet (ALPHA)
PR-002	Parser TCAC scellé et immuable	Complet pour la fixture de test
PR-003	Moteur de règles ADÈLE minimal	Complet
PR-004	Format de preuve METATRON	Complet pour le hachage d'intégrité de test
PR-005	Journal en ajout seul	Complet
PR-006	Merkle et rapport SAK simple	Complet
PR-007	Vérificateur Rust indépendant	Complet
PR-008	API sandbox SVP-X Evidence/Audit	Complet
PR-009	Explicabilité ADÈLE et scanner de contradictions	Complet
PR-010	Profils SAK v2 et rédaction	Complet
PR-011	Spécification et mock supervisé Némésis	Complet
PR-012	Démonstration Longueuil reproductible	Complet

Limites explicites de l'état actuel

- Aucune validation matérielle, terrain, juridique, institutionnelle ou réglementaire n'a été conduite.
- Aucun audit de sécurité externe indépendant n'a été réalisé sur cette base de code.
- Les signatures demeurent des valeurs de test (placeholders) ; aucune cryptographie de production n'est utilisée.
- Némésis ne dépasse pas le stade de mock supervisé.
- Cette livraison logicielle ALPHA n'est ni une certification, ni une validation de déploiement réel.

Engagement de langage

Le protocole s'interdit d'employer les termes `irréfutable`, `preuve absolue`, `inviolable` ou `Production-Proven` pour qualifier son état présent. Cet engagement fait partie des règles non négociables du projet, au même titre que la séparation Tier 1 / Tier 2.

Contribuer ou évaluer le protocole

La spécification, les principes de gouvernance et l'état d'avancement détaillé sont documentés pour permettre un examen technique indépendant. Pour toute question d'architecture ou pour proposer une revue externe, contactez l'équipe SerenityVault.

`info@serenityvault.org · protocole.org`

SerenityVault Protocol, Série 8 — Document technique. Décrit une architecture logicielle au stade ALPHA ; ne constitue ni une spécification finale, ni une certification de sécurité.